(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 439 464 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
21.07.2004 Bulletin 2004/30

(51) Int Cl.[7]: **G06F 9/44**

(21) Application number: **03256913.9**

(22) Date of filing: **31.10.2003**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PT RO SE SI SK TR
Designated Extension States:
AL LT LV MK

(30) Priority: **14.11.2002 US 295525**

(71) Applicant: **Sun Microsystems, Inc.**
Santa Clara, CA 95054 (US)

(72) Inventors:
- **Flynn, Martin D.**
  Grass Valley, California 95949 (US)
- **Mehta, Chinmay S.**
  San Jose, California 95134 (US)

(74) Representative: **Collins, John David**
Marks & Clerk,
57-60 Lincoln's Inn Fields
London WC2A 3LS (GB)

(54) Java interface for accessing graphical user interface-based java tools

(57) Techniques for accessing Java-based tools are disclosed. The Java tools provide a Graphical User Interface (GUI) which can be used to perform various operations in a Java computing environment. Typically, these operations are performed by a user with respect to one or more Java components. A Java programming interface is disclosed. The Java programming interface can be used to access the graphical user interface of a Java tool. As such, the Java programming interface can be used to simulate operations that can be performed using the GUI of the Java tool. This allows implementation of many programs for various applications. These applications, for example, include automated testing and tutorials for the GUI-based tools.

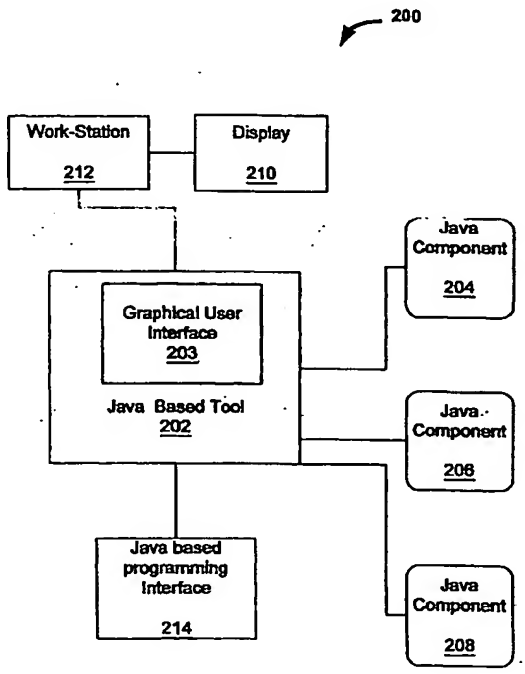

Fig. 2

EP 1 439 464 A2

Description

BACKGROUND OF THE INVENTION

[0001] The present invention relates generally to Java computing environments and, more particularly, to techniques for accessing Java tools which operate in Java computing environments.

[0002] One of the goals of high level languages is to provide a portable programming environment such that the computer programs may easily be ported to another computer platform. High level languages such as "C" provide a level of abstraction from the underlying computer architecture and their success is well evidenced from the fact that most computer applications are now written in a high level language.

[0003] Portability has been taken to new heights with the advent of the World Wide Web ("the Web"), which is an interface protocol for the Internet which allows communication of diverse computer platforms through a graphical interface. Computers communicating over the Web are able to download and execute small applications called applets. Given that applets may be executed on a diverse assortment of computer platforms, the applets are typically executed by a Java™ virtual machine.

[0004] Recently, the Java programming environment has become quite popular. The Java programming language is a language that is designed to be portable enough to be executed on a wide range of computers ranging from small devices (e.g., pagers, cell phones and smart cards) up to supercomputers. Computer programs written in the Java programming language (and other languages) may be compiled into Java Bytecode instructions that are suitable for execution by a Java virtual machine implementation. The Java virtual machine is commonly implemented in software by means of an interpreter for the Java virtual machine instruction set but, in general, may be software, hardware, or both. A particular Java virtual machine implementation and corresponding support libraries together constitute a Java runtime environment.

[0005] Computer programs in the Java programming language are arranged in one or more classes or interfaces (referred to herein jointly as classes or class files). Such programs are generally platform, i.e., hardware and operating system, independent. As such, these computer programs may be executed without modification on any computer that is able to run an implementation of the Java runtime environment.

[0006] Object-oriented classes written in the Java programming language are compiled to a particular binary format called the "class file format." The class file includes various components associated with a single class. These components can be, for example, methods and/or interfaces associated with the class. In addition, the class file format can include a significant amount of ancillary information that is associated with the class. The class file format (as well as the general operation of the Java virtual machine) is described in some detail in The Java Virtual Machine Specification, Second Edition, by Tim Lindholm and Frank Yellin.

[0007] Fig. 1A shows a progression of a simple piece of Java source code through execution by an interpreter, the Java virtual machine. Java source code 101 includes the classic Hello World program written in Java. The source code is then input into a Bytecode compiler 103 that compiles the source code into Bytecodes. The Bytecodes are virtual machine instructions as they will be executed by a software emulated computer. Typically, virtual machine instructions are generic (*i.e.*, not designed for any specific microprocessor or computer architecture) but this is not required. The Bytecode compiler outputs a Java class file 105 that includes the Bytecodes for the Java program. The Java class file is input into a Java virtual machine 107. The Java virtual machine is an interpreter that decodes and executes the Bytecodes in the Java class file. The Java virtual machine is an interpreter, but is commonly referred to as a virtual machine as it emulates a microprocessor or computer architecture in software (e.g., the microprocessor or computer architecture may not exist in hardware).

[0008] The Java programming environment can provide a component-based environment. For example, the Java 2 Platform Enterprise Edition (J2EE) developed by Sun Microsystems provides a component-based approach to the design, development, assembly and deployment of enterprise applications. The J2EE platform stresses the practical aspects of enterprise development by minimizing the need to modify code in order to deploy applications. On the server side, the J2EE platform includes two major containers: the web container and the Enterprise JavaBeans (EJB) server. The web container provides the functionality to support servlet and JavaServer Pages (JSP) technology. The EJB server provides the underpinnings to support Enterprise JavaBeans technology.

[0009] The component-based environment of the J2EE platform can be a complex environment with many components. As such, a deployment tool has been developed which can perform the actual installation of the EJB components and the additional classes and interfaces into the EJB container operational environment. The Deployer is an expert at a specific operational environment and is responsible for the deployment of EJB components. The Deployer provides a GUI which can be used to conveniently perform tasks. However, even with a sophisticated GUI-based tool, assembling and deploying an EJB technology-based application can be a complex process. Moreover, with the growing complexity of the Java environment, the task of testing and/or using the deployment tool itself has become an important issue.

[0010] Accordingly, techniques for accessing GUI-based Java tools are needed.

SUMMARY OF THE INVENTION

[0011] Broadly speaking, the present invention relates to techniques for accessing GUI based Java tools. These tools provide a Graphical User Interface (GUI) which can be used to perform various operations in Java computing environments. Typically, a user interacts with the GUI to perform these operations with respect to one or more java components.

[0012] In accordance with one aspect of the invention, a Java programming interface is disclosed. The Java programming interface can be used to access the GUI of a Java tool. As such, the Java programming interface can be used to simulate operations that can be performed by users who interact with the GUI of a Java tool. This allows implementations of many programs for various applications. These applications, for example, include automated testing suites which can be used to test the GUI-based tools, as well as tutorials which can be provided for the users of these tools.

[0013] The invention can be implemented in numerous ways, including as a method, an apparatus, a computer readable medium, and a database system. Several embodiments of the invention are discussed below.

[0014] One embodiment of the invention can be implemented as a Java Programming Interface capable of interacting with a Java GUI-based tool in a Java computing environment. The Java GUI-based tool provides a GUI which can be used to perform one or more operations relating to one or more components operating in the Java computing environment. Furthermore, the Java Programming Interface provides at least one functionality which can be used to perform at least one operation associated with interaction with the GUI of the Java GUI-based tool. It should be noted that the at least one operation can also be performed by interacting with said GUI of said Java GUI-based tool.

[0015] Another embodiment of the invention can be implemented as a computing system which is capable of using a Java Programming Interface suitable for interaction with a Java deployment tool in a Java computing environment. The Java deployment tool provides a GUI which can be used to perform one or more operations relating to one or more components in the Java computing environment. it should be noted that the one or more operations include deployment of said one or more components in said Java computing environment and the Programming Interface provides at least one functionality which can be used to perform at least one operation relating to deployment of the one or more components. It should also be noted that at least one operation can also be performed by a user of said Java deployment tool by interacting with said GUI.

[0016] As a computer readable medium including computer program code for a Java Programming Interface, one embodiment of the invention can be used to interact with a Java deployment tool in a Java computing environment. The Java deployment tool provides a GUI which can be used to perform one or more operations relating to one or more components in the Java computing environment. The one or more operations include deployment of the one or more components in the Java computing environment and the Java Programming Interface provides at least one functionality which can be used to perform at least one operation relating to deployment of the one or more components. Again, it should also be noted that at least one operation can also be performed by a user of said Java deployment tool by interacting with said GUI.

[0017] These and other aspects and advantages of the present invention will become more apparent when the detailed description below is read in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

Fig. 1 shows a progression of a simple piece of Java source code through execution by an interpreter, the Java virtual machine.

Fig. 2 illustrates a Java computing environment including a Java-based tool in accordance with one embodiment of the invention.

Fig. 3 illustrates a Java computing environment including a Java deployment tool in accordance with one embodiment of the invention.

Fig. 4 depicts an exemplary sequence of screens which can be generated using the Java-based programming interface in accordance with one embodiment of the invention.

Fig. 5A depicts an exemplary screen in accordance with one embodiment of the invention.

Fig. 5B depicts an exemplary screen in accordance with another embodiment of the invention.

Fig. 6A-B illustrate programming scripts in accordance with various embodiments of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0019] As noted in the background, techniques for accessing GUI-based Java tools are needed. Accordingly, the present invention pertains to techniques for accessing GUI-based Java tools. These tools provide a Graphical User Interface (GUI) which can be used to perform various operations in a Java computing environment. Typically, a user interacts with the GUI to perform these operations with respect to one or more Java components.

[0020] In accordance with one aspect of the invention, a Java programming interface is disclosed. The Java programming interface can be used to access the GUI of a Java tool. As such, the Java programming interface can be used to simulate operations that can be performed by users who interact with the GUI of a Java tool. This allows implementations of many programs for various applications. These applications, for example, include automated testing suites which can be used to test the GUI-based tools, as well as tutorials which can be provided for the users of these tools.

[0021] Embodiments of the invention are discussed below with reference to Figs. 2-6B. However, those skilled in the art will readily appreciate that the detailed description given herein with respect to these figures is for explanatory purposes only as the invention extends beyond these limited embodiments.

[0022] Fig. 2 illustrates a Java computing environment 200 including a Java-based tool 202 in accordance with one embodiment of the invention. The Java-based tool 202 provides a Graphical User Interface (GUI) 203 which allows a user to conveniently access its functionality. Typically, the Java-based tool 202 can be used to perform tasks related to one or more Java components 204, 206 and 208. The Graphical User Interface (GUI) 203 of the Java-based tool 202 can operate to display information on a display 210 associated with a work-station 212. In this way, the user can perform tasks by interacting with the Graphical User Interface 203 which is displayed on the display 210.

[0023] As shown in Fig. 2, a Java-based Programming Interface (PI) 214 can be provided in accordance with one embodiment of the invention. As will be appreciated, the Programming Interface (PI) 214 provides an interface to the Java-based tool 202 which can be used to perform a variety of tasks related to the Graphical User Interface 203. These tasks, for example, include writing programs which can be used as a test or a tutorial for the Java-based tool 202.

[0024] Fig. 3 illustrates a Java computing environment 300 including a Java deployment tool 302 in accordance with one embodiment of the invention. The deployment tool 302 can perform a variety of tasks in the Java computing environment 300. These tasks, for example, include the installation of: an EJB component 306, a client application 308, resource adapters 310 and Web applications (JSP Servlets) 312. It should be noted that the deployment tool 302 can interact with Java classes and/or interfaces associated with these components. As such, the Java deployment tool 302 is an expert tool in the specific operational environment 300 and is responsible for the deployment of various Java components which operate in this environment. During operation, the deployment tool 302 can generate one or more output files which typically reside on a server 314 (e.g., Web Archive file (WAR) 316, Java Archive file (JAR) 318, Resource Archive file (RAR) 320, Enterprise Archive file (EAR) 322, etc.).

[0025] As illustrated in Fig. 3, a Java-based programming interface 330 has been provided for the Java deployment tool 302. The Java-based programming interface 330 includes a Java application layer 332 and a Java system layer 334. As will be appreciated, the two layer design of the Java-based programming interface 330 provides a level of abstraction which allows the application layer to hide the complexities associated with performing tasks at the system level.

[0026] As shown in Fig. 3, the Java system layer 334 interacts with the GUI 336 of the Java deployment tool 302. As such, the Java-based programming interface 330 can be used to develop programs capable of performing a variety of tasks. These tasks, for example, include developing test scripts and tutorials for the Java deployment tool 302. The test scripts can be used, for example, to provide an automated environment where the Java deployment tool 302 can be conveniently tested. Similarly, the Java-based programming interface 330 can be used to develop programming scripts for tutorials. These tutorials can be used to educate the user about the Java deployment tool 302. This can be achieved, for example, by providing programming scripts that can show the user how to perform a given task. Moreover, the user can step through the tutorials at his or her own pace. This allows novice users to learn about the capabilities of the Java deployment tool 302 step by step while reading the useful information that is provided.

[0027] As noted above, the Java-based programming interface 330 interacts with the GUI 336 of the Java deployment tool 302. Typically, performing a task using the Java deployment tool 302 requires the user to interact with one or more screens which are generated by the GUI 336. Accordingly, the Java-based programming interface 304 can be used to simulate the user interactions with the GUI 336. This simulation can be used, for example, to test the Java deployment tool 302. Furthermore, the Java-based programming interface 330 can be used to provide input to the GUI 336 so that particular screens are presented to the user at desired times. Similarly, based on user input, a desired action can be

taken. For example, after the user selects a "continue" option, a new screen can be generated. As another example, when the user selects a "help" option, information regarding the screen that is currently being displayed can be provided. In this way, the Java-based programming interface 330 can be used to provide an interactive controlled environment which can, for example, be used to provide useful tutorials for the Java deployment tool 302.

[0028] In any case, user interaction with the GUI 336 can be represented by a series of screens. To illustrate, Fig. 4 depicts an exemplary sequence of screens 400 which can be generated using the Java-based programming interface 304 in accordance with one embodiment of the invention. The exemplary sequence of screens 400 represents the screens which are generated when one or more tasks are performed using a GUI of a GUI-based Java tool (e.g., using the Java deployment tool 302 via the GUI 336). As such, the exemplary sequence of screens 400 can, for example, represent a series of screens which are generated when a test script or tutorial script is executed.

[0029] Initially, a screen 402 is generated. The screen 402 can, for example, be general screen (or tab). Next, a screen 404 is generated. It should be noted that screen 404 is one of the screens that may be generated from screen 402. In other words, screen 406 or 408 may be selected in another programming sequence. Similarly, a screen 410 is generated after generation of screen 404. Thereafter, screens 412 and 414 are generated in the sequence 400.

[0030] To further illustrate, Fig. 5A depicts an exemplary screen 500 in accordance with one embodiment of the invention. The exemplary screen 500 can, for example, represent a screen in the sequence of screens 400 of Fig. 4. It should be noted that the exemplary screen 500 can, for example, be generated using the Java-based programming interface 304 of Fig. 3. As shown in Fig. 5A, screen 500 includes display portions 502, 504, and 506 representing various fields associated with the screen 500. As will be appreciated, these fields can be set to various values using the Java-based programming interface 304. As such, various testing scenarios can be developed to test the Java deployment tool 302. Fig. 5B depicts an exemplary screen 510 in accordance with another embodiment of the invention. As shown in Fig. 5B, screen 510 includes selectable options 512 and 514 which allow the user to interact with a programming script during execution. By way of example, the pause/continue option 512 can be selected by the user to pause/continue the execution of a tutorial which is developed for the Java deployment tool 302. Similarly, selection of the help option 514 can provide the user with information regarding a desired topic. In addition, general information regarding the screen 510 can be displayed in display portion 516 during the tutorial.

[0031] As illustrated in Fig. 3, the Java-based programming interface 330 includes an application layer 332 and a system layer 334. As noted above, the two layer design of the Java-based programming interface 330 provide a level of abstraction which allows the application layer to hide the complexities associated with performing tasks at the system level.

[0032] To elaborate, Fig. 6A illustrates a programming script 600 in accordance with one embodiment of the invention. The programming script 600 can be written, for example, using the Java-based programming interface 330 to interact with the GUI 336 of the deployment tool 302 shown in Fig. 3. As will be appreciated, the programming script 600 hides the complexities associated with performing tasks at the system level. Accordingly, it is possible to use high level functions such as finding a particular screen, finding a particular field in that screen and entering a specified value in that field. For example, a series of high level functions 602, 604 and 606 can respectively be used to find a particular screen, find a particular field in that screen, and enter a particular value in that field.

[0033] It should be noted that it is also possible to reduce the number of operations need to write a programming script. For example, if the arrangement of the fields in the screen are known and/or remains constant, there is no need to find particular fields in the screen. As such, as illustrated in programming script 650 of Fig. 6B, the values for the fields in the screen can be entered in a sequence of high level functions 620, 622, 624 and 626 which respectively find a screen A and serially enter the values X, Y and Z in the fields of the screen A.

[0034] In accordance with one embodiment of the invention disclosed hereinafter are exemplary Java interfaces which can be used to implement a class "script runner" and a class "Ejb Wizard". The class "script runner" represents a system level interface. The class "Ejb Wizard represents an application level interface. As will be appreciated by those skilled in the art, these classes can be used to provide a Java programming interface for a Java deployment tool.

com.sun.enterprise.tools.deployment.ui.script

Class ScriptRunner

```
java.lang.Object
  |
  +--com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore
        |
        +--com.sun.enterprise.tools.deployment.ui.script.ScriptRunner
```

Direct Known Subclasses:

Deploytool, J2eeComponent

---

public class **ScriptRunner**

extends ScriptRunnerCore

Inner classes inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore
ScriptRunnerCore.ModalWindowProbe

Field Summary

protected static java.lang.String	CANCEL
static boolean	CheckActiveWindow
protected static java.lang.String	FINISH
protected static java.lang.String	NEXT

Fields inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore
debug, EDIT_CONTENTS, MAIN_WINDOW

Constructor Summary

ScriptRunner()

Method Summary

void	clickButton(java.awt.Component p, java.lang.String t)
void	closeAllWindows()
void	expandTreeItem(java.awt.Component p, java.lang.String t, java.lang.String v)

void	selectCheckBoxList(java.awt.Component p, java.lang.String t, java.lang.String[] v)
void	selectComboBox(java.awt.Component p, java.lang.String t, java.lang.String v)
void	selectMenuItem(java.lang.String[] tree)
void	selectRadioButton(java.awt.Component p, java.lang.String t)
void	selectTab(java.lang.String t)
void	selectTableRow(java.awt.Component p, java.lang.String title, java.lang.String rowVal)
void	selectTableRow(java.awt.Component p, java.lang.String title, java.lang.String[] rowVal)
void	selectTreeDescriptor(java.lang.String d)
void	selectTreeDescriptor(java.lang.String[] d)
void	selectTreeItems(java.awt.Component p, java.lang.String t, java.lang.String[] v)
void	setCheckBox(java.awt.Component p, java.lang.String t, boolean value)
void	setTableValues(java.awt.Component p, java.lang.String title, java.lang.String[] rowVal, java.lang.String[] val)
void	setTableValues(java.awt.Component p, java.lang.String title, java.lang.String rowVal, java.lang.String[] val)
void	setTextArea(java.awt.Component p, java.lang.String t, java.lang.String v)
void	setTextField(java.awt.Component p, java.lang.String t, java.lang.String v)
Methods inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore	
findWindow, addWindowActivationListener, compareTitle, debugMsg, error, findActiveWindow, findComponent, findComponent, findModalWindow, findWindow, findWindow, findWindow, getComponentTitle, getComponentTitle, getComponentTitle, getDepthFirstWindowList, getScriptSource, indent, invoke, invoke, isWindowActive, pause, printWindows, removeWindowActivationListener	
Methods inherited from class java.lang.Object	
clone, equals, finalize, getClass, hashCode, notify, notifyAll.	

toString, wait, wait, wait

Field Detail

NEXT

```
protected static final java.lang.String NEXT
```

CANCEL

```
protected static final java.lang.String CANCEL
```

FINISH

```
protected static final java.lang.String FINISH
```

CheckActiveWindow

```
public static boolean CheckActiveWindow
```

Constructor Detail

ScriptRunner

```
public ScriptRunner()
```

Method Detail

closeAllWindows

```
public void closeAllWindows()
                    throws ScriptException
```

clickButton

```
public void clickButton(java.awt.Component p,
                        java.lang.String t)
                 throws ScriptException
```

selectRadioButton

```
public void selectRadioButton(java.awt.Component p,
                              java.lang.String t)
                       throws ScriptException
```

selectComboBox

```
public void selectComboBox(java.awt.Component p,
                           java.lang.String t,
                           java.lang.String v)
                    throws ScriptException
```

setCheckBox

```
public void setCheckBox(java.awt.Component p,
                        java.lang.String t,
```

```
                        boolean value)
                 throws ScriptException
```

---

setTextField

```
public void setTextField(java.awt.Component p,
                         java.lang.String t,
                         java.lang.String v)
                  throws ScriptException
```

---

setTextArea

```
public void setTextArea(java.awt.Component p,
                        java.lang.String t,
                        java.lang.String v)
                 throws ScriptException
```

---

selectCheckBoxList

```
public void selectCheckBoxList(java.awt.Component p,
                               java.lang.String t,
                               java.lang.String[] v)
                        throws ScriptException
```

---

selectTreeItems

```
public void selectTreeItems(java.awt.Component p,
                            java.lang.String t,
                            java.lang.String[] v)
                     throws ScriptException
```

---

expandTreeItem

```
public void expandTreeItem(java.awt.Component p,
                           java.lang.String t,
                           java.lang.String v)
                    throws ScriptException
```

---

selectTreeDescriptor

```
public void selectTreeDescriptor(java.lang.String d)
                          throws ScriptException
```

---

selectTreeDescriptor

```
public void selectTreeDescriptor(java.lang.String[] d)
                          throws ScriptException
```

---

selectTableRow

```
public void selectTableRow(java.awt.Component p,
                           java.lang.String title,
                           java.lang.String rowVal)
                    throws ScriptException
```

---

selectTableRow

```
public void selectTableRow(java.awt.Component p,
                           java.lang.String title,
```

```
                                java.lang.String[] rowVal)
                        throws ScriptException
```

---

setTableValues

```
public void setTableValues(java.awt.Component p,
                           java.lang.String title,
                           java.lang.String rowVal,
                           java.lang.String[] val)
                    throws ScriptException
```

---

setTableValues

```
public void setTableValues(java.awt.Component p,
                           java.lang.String title,
                           java.lang.String[] rowVal,
                           java.lang.String[] val)
                    throws ScriptException
```

---

selectTab

```
public void selectTab(java.lang.String t)
                throws ScriptException
```

---

selectMenuItem

```
public void selectMenuItem(java.lang.String[] tree)
                    throws ScriptException
```

Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.sun.enterprise.tools.deployment.ui.script

Class EjbWizard

```
java.lang.Object
  |
  +--com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore
        |
        +--com.sun.enterprise.tools.deployment.ui.script.ScriptRunner
              |
              +--
com.sun.enterprise.tools.deployment.ui.script.J2eeComponent
                    |
                    +--
com.sun.enterprise.tools.deployment.ui.script.J2eeWizard
                          |
                          +--
com.sun.enterprise.tools.deployment.ui.script.EjbWizard
```

All Implemented Interfaces:

GuiConstants

---

public class **EjbWizard**
extends J2eeWizard
This class consists of methods that are specific to the EJB wizard.

Inner classes inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore
ScriptRunnerCore.ModalWindowProbe
Fields inherited from class com.sun.enterprise.tools.deployment.ui.script.J2eeComponent
win
Fields inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunner
CANCEL, CheckActiveWindow, FINISH, NEXT
Fields inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore
debug, EDIT CONTENTS, MAIN WINDOW
Fields inherited from interface com.sun.enterprise.tools.deployment.ui.script.GuiConstants
ADD RELATIONSHIP DIALOG, ADD USER DIALOG, AddButton, AddUserButton, APPCLIENTWIZARD WINDOW, BASE DIR1, CERTIFICATE FILE DIALOG, CONFIGINSTALL DIALOG, CONFIRMATION DIALOG, DEPLOYMENT SETTINGS DIALOG, EAR, EDIT CONTENTS WINDOW, EJB REFS TAB, EJBWIZARD WINDOW, ERROR DIALOG, EVENT LISTENERS TAB, FINDERSELECT METHODS WINDOW, GENERAL TAB, GROUPS DIALOG, INFORMATION DIALOG, JNDI NAMES TAB, NEWAPP WINDOW, OKButton, RAR, RARWIZARD WINDOW, RELATIONSHIPS TAB, RESOURCE REFS TAB, SERVER, ServerConfigMenu. ToolsMenu. WARNING DIALOG. WARWIZARD WINDOW.

WEB CONTEXT ROOT TAB, WIZARD WINDOW

Constructor Summary

`EjbWizard(Deploytool dtw, java.lang.String baseDir)` This constructor invokes the EJB wizard and initializes the Script base directory and an instance of the ScriptRunner for the ContentsEditor class.

Method Summary

void	`addToExistingEjbFile(java.lang.String jarDisplayName, java.lang.String appName)` This method selects the "Add to Existing JAR File" radio button on the EJB JAR screen and selects the value formed by concatenating the specified parameters in a predetermined way from the corresponding combo box.
void	`createNewEjbFile(java.lang.String jarDisplayName)` This method selects the "Create New JAR File in Application" radio button on the EJB JAR screen and sets the JAR Display Name to the value specified by the jarDisplayName parameter.
void	`selectEnterpriseBeanClass(java.lang.String className)` This method selects the specified class name for the "Enterprise Bean Class" combo box on the General screen of the New EJB wizard.
void	`selectEnterpriseBeanName(java.lang.String name)` This method sets the "Enterprise Bean Name" on the General screen of the New EJB wizard to the value specified by the name parameter.
void	`selectLocalHomeInterface(java.lang.String localHome)` This method selects the specified interface class name for the "Local Home Interface" combo box on the General screen of the New EJB wizard.
void	`selectLocalInterface(java.lang.String local)` This method selects the specified interface class name for the "Local Interface" combo box on the General screen of the New EJB wizard.
void	`selectRemoteHomeInterface(java.lang.String remoteHome)` This method selects the specified interface class name for the "Remote Home Interface" combo box on the General screen of the New EJB wizard.
void	`selectRemoteInterface(java.lang.String remote)` This method selects the specified interface class name for the "Remote Interface" combo box on the General screen of the New EJB wizard.
void	`setBeanTypeEntity()` This method selects the "Entity" radio button on the General screen of the New EJB wizard.
void	`setBeanTypeMessage()` This method selects the "Message-Driven" radio button on the General screen of the New EJB wizard.
void	`setBeanTypeStatefulSession()` This method selects the "Session" and "Stateful" radio buttons on the General screen of the New EJB wizard.
void	`setBeanTypeStatelessSession()` This method selects the "Session" and "Stateless" radio buttons on the General

	screen of the New EJB wizard.

Methods inherited from class com.sun.enterprise.tools.deployment.ui.script.J2eeWizard
finishWizard, nextScreen

Methods inherited from class com.sun.enterprise.tools.deployment.ui.script.J2eeComponent
addAlias, addArchiveFiles, addEjbReference, addEnvironmentEntry, addResourceEnvReference, addResourceReference, addServletFilter, addServletFilterMapping, checkDurableSubscriber, checkPersistentFields, invokeContentsEditor, selectAutoAcknowledgement, selectConnectionFactory, selectDestination, selectDupOKAcknowledgement, setAbstractSchemaName, setCMP20, setDestinationTypeQueue, setDestinationTypeTopic, setEnterpriseBeanName, setJmsMsgSelector, setPrimaryKeyClass, setPrimaryKeyField, setSqlLocal, setSqlRemote

Methods inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunner
clickButton, closeAllWindows, expandTreeItem, selectCheckBoxList, selectComboBox, selectMenuItem, selectRadioButton, selectTab, selectTableRow, selectTableRow, selectTreeDescriptor, selectTreeDescriptor, selectTreeItems, setCheckBox, setTableValues, setTableValues, setTextArea, setTextField

Methods inherited from class com.sun.enterprise.tools.deployment.ui.script.ScriptRunnerCore
findWindow, addWindowActivationListener, compareTitle, debugMsg, error, findActiveWindow, findComponent, findComponent, findModalWindow, findWindow, findWindow, findWindow, getComponentTitle, getComponentTitle, getComponentTitle, getDepthFirstWindowList, getScriptSource, indent, invoke, invoke, isWindowActive, pause, printWindows, removeWindowActivationListener

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EjbWizard

```
public EjbWizard(Deploytool dtw,
                 java.lang.String baseDir)
          throws ScriptException
```

This constructor invokes the EJB wizard and initializes the Script base directory and an instance of the ScriptRunner for the ContentsEditor class.

Parameters:

`dtw` - The handle to the Deploytool main window.

`baseDir` - The Script base directory.

Throws:

`ScriptException` - The Script Exception.

Method Detail

createNewEjbFile

```
public void createNewEjbFile(java.lang.String jarDisplayName)
                      throws ScriptException
```

This method selects the "Create New JAR File in Application" radio button on the EJB JAR screen and sets the JAR Display Name to the value specified by the jarDisplayName parameter.

Parameters:

`jarDisplayName` - The display name for the new jar file.

Throws:

`ScriptException` - The Script Exception.

---

addToExistingEjbFile

```
public void addToExistingEjbFile(java.lang.String jarDisplayName,
                                 java.lang.String appName)
                          throws ScriptException
```

This method selects the "Add to Existing JAR File" radio button on the EJB JAR screen and selects the value formed by concatenating the specified parameters in a predetermined way from the corresponding combo box.

Parameters:

`jarDisplayName` - The display name of the existing jar file.

`appName` - The name of the current application.

Throws:

`ScriptException` - The Script Exception.

---

setBeanTypeEntity

```
public void setBeanTypeEntity()
                      throws ScriptException
```

This method selects the "Entity" radio button on the General screen of the New EJB wizard.

Throws:

`ScriptException` - The Script Exception.

---

setBeanTypeStatelessSession

```
public void setBeanTypeStatelessSession()
                                 throws ScriptException
```

This method selects the "Session" and "Stateless" radio buttons on the General screen of the New EJB wizard.

Throws:

`ScriptException` - The Script Exception.

---

setBeanTypeStatefulSession

```
public void setBeanTypeStatefulSession()
                                 throws ScriptException
```

This method selects the "Session" and "Stateful" radio buttons on the General screen of the New EJB wizard.

Throws:

`ScriptException` - The Script Exception.

---

setBeanTypeMessage

```
public void setBeanTypeMessage()
                        throws ScriptException
```

This method selects the "Message-Driven" radio button on the General screen of the New EJB wizard.

Throws:

`ScriptException` - The Script Exception.

---

selectEnterpriseBeanClass

```
public void selectEnterpriseBeanClass(java.lang.String className)
                                throws ScriptException
```

This method selects the specified class name for the "Enterprise Bean Class" combo box on the General screen of the New EJB wizard.

Parameters:

`className` - The Enterprise Bean class.

Throws:

`ScriptException` - The Script Exception.

---

selectEnterpriseBeanName

```
public void selectEnterpriseBeanName(java.lang.String name)
                               throws ScriptException
```

This method sets the "Enterprise Bean Name" on the General screen of the New EJB wizard to the value specified by the name parameter.

Parameters:

`name` - The Enterprise Bean Name.

Throws:

`ScriptException` - The Script Exception.

---

selectLocalHomeInterface

```
public void selectLocalHomeInterface(java.lang.String localHome)
                               throws ScriptException
```

This method selects the specified interface class name for the "Local Home Interface" combo box on the General screen of the New EJB wizard.

Parameters:

`localHome` - The Local Home interface for the bean.

Throws:

`ScriptException` - The Script Exception.

---

selectLocalInterface

```
public void selectLocalInterface(java.lang.String local)
                           throws ScriptException
```

This method selects the specified interface class name for the "Local Interface" combo box on the General screen of the New EJB wizard.

Parameters:

`local` - The Local interface for the bean.

Throws:

`ScriptException` - The Script Exception.

selectRemoteHomeInterface

```
public void selectRemoteHomeInterface(java.lang.String remoteHome)
                                throws ScriptException
```

This method selects the specified interface class name for the "Remote Home Interface" combo box on the General screen of the New EJB wizard.

Parameters:

`remoteHome` - The Remote Home interface for the bean.

Throws:

`ScriptException` - The Script Exception.

selectRemoteInterface

```
public void selectRemoteInterface(java.lang.String remote)
                            throws ScriptException
```

This method selects the specified interface class name for the "Remote Interface" combo box on the General screen of the New EJB wizard.

Parameters:

`remote` - The Remote interface for the bean.

Throws:

`ScriptException` - The Script Exception.

Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Claims

1. In a Java computing environment, a Java Programming Interface capable of interacting with a Java GUI-based tool,
   wherein said Java GUI-based tool provides a GUI which can be used to perform one or more operations relating to one or more components operating in said Java computing environment;
   wherein said Java Programming Interface provides at least one functionality which can be used to perform at least one operation associated with interaction with said GUI of said Java GUI-based tool; and
   wherein said at least one operation can also be performed by interacting with said GUI of said Java GUI-based tool.

2. A Java Programming Interface as recited in claim 1, wherein said Java Programming interface is capable of being used to write programming scripts.

3. A Java Programming Interface as recited in claim 1, wherein said programming script is a test script which can be used to test said Java GUI-based tool.

4. A Java Programming Interface as recited in claim 1, wherein said programming script is a tutorial script for said GUI-based tool.

5. A Java Programming Interface as recited in claim 1,
   wherein said Java GUI-based tool is a Java deployment tool; and
   wherein said one or more Java components are one or more WEB applications.

6. A Java Programming Interface as recited in claim 1,
wherein said Java GUI-based tool is a Java deployment tool; and
wherein said one or more Java components can be one or more of the following components: an Enterprise Java Bean, a client application, a resource adaptor, and a WEB application.

7. A Java Programming Interface as recited in claim 1, wherein said Java Programming Interface comprises:

a Java-based Programming Interface which can be used to simulate interactions with said Java GUI-based deployment tool;
a Java-based system layer which can operate to implement said one or more interactions with said at least one screen.

8. A Java Programming Interface as recited in claim 1, wherein said at least one functionality is an operation which can be performed by a user when interacting with a screen generated by said GUI of said GUI-based tool.

9. A Java Programming Interface as recited in claim 8, wherein said at least one functionality is finding a screen or field in a screen, or entering a value in a field in a screen, said screen being capable of being generated by said GUI of said Java-based tool.

10. In a Java computing environment, a Java Programming Interface capable of interacting with a Java deployment tool,
wherein said Java deployment tool provides a GUI which can be used to perform one or more operations relating to one or more components in said Java computing environment; said one or more operations including deployment of said one or more components in said Java computing environment;
wherein said Java Programming Interface provides at least one functionality which can be used to perform at least one operation relating to deployment of said one or more components;
wherein said at least one operation can also be performed by a user of said Java deployment tool by interacting with said GUI.

11. A Java Programming Interface as recited in claim 10, wherein said Java Programming Interface is capable be being used to write programming scripts.

12. A Java Programming Interface as recited in claim 11, wherein said programming script is a test script which can be used to test said Java deployment tool.

13. A Java Programming Interface as recited in claim 11, wherein said programming script is a tutorial script for said Java deployment tool.

14. A Java Programming Interface as recited in claim 11,
wherein said one or more Java components are one or more WEB applications.

15. A Java Programming Interface as recited in claim 11, wherein said one or more Java components can be one or more of the following components: an Enterprise Java Bean, a client application, a resource adaptor, and a web application.

16. A Java Programming Interface as recited in claim 11, wherein said Java Programming Interface comprises:

a Java-based Programming Interface which can be used to simulate interactions with one or more screens which can be generated using said GUI of said Java deployment tool; and
a Java-based system layer which can operate to implement said one or more interactions.

17. A Java Programming Interface as recited in claim 16 wherein said at least one functionality is finding a screen or field in a screen, or entering a value in a field in a screen, said screen being capable of being generated by said GUI of said Java deployment tool.

18. A computing system capable of using a Java Programming Interface which can be used to interact with a Java deployment tool in a Java computing environment;
wherein said Java deployment tool provides a GUI which can be used to perform one or more operations

relating to one or more components in said Java computing environment; said one or more operations including deployment of said one or more components in said Java computing environment;

wherein said Java Programming Interface provides at least one functionality which can be used to perform at least one operation relating to deployment of said one or more components;

wherein said at least one operation can also be performed by a user of said Java deployment tool by interacting with said GUI.

19. A computing system as recited in claim 18, wherein said Java Programming interface is capable of being used to write programming scripts.

20. A computing system as recited in claim 18, wherein said programming script is a test script which can be used to test said Java deployment tool.

21. A computing system as recited in claim 18, wherein said programming script is a tutorial script for said Java deployment tool.

22. A computing system as recited in claim 18, wherein said one or more Java components are one or more WEB applications.

23. A computing system as recited in claim 18, wherein said one or more Java components can be one or more of the following components: an Enterprise Java Bean, a client application, a resource adaptor, and a web application.

24. A computing system as recited in claim 18, wherein said Java Programming Interface comprises:

a Java-based Programming Interface which can be used to simulate interactions with one or more screens which can be generated using said GUI of said Java deployment tool; and
a Java-based system layer which can operate to implement said one or more interactions.

25. A computing system as recited in claim 18, wherein said at least one functionality is finding a screen or field in a screen, or entering a value in a field in a screen, said screen being capable of being generated by said GUI of said Java deployment tool.

26. A computer readable medium including computer program code for a Java Programming Interface which can be used to interact with a Java deployment tool in a Java computing environment;

wherein said Java deployment tool provides a GUI which can be used to perform one or more operations relating to one or more components in said Java computing environment; said one or more operations including deployment of said one or more components in said Java computing environment;

wherein said Java Programming Interface provides at least one functionality which can be used to perform at least one operation relating to deployment of said one or more components;

wherein said at least one operation can also be performed by a user of said Java deployment tool by interacting with said GUI.

27. A computer readable medium as recited in claim 26, wherein said Java Programming Interface is capable of being used to write programming scripts.

28. A computer readable medium as recited in claim 26, wherein said programming script is a test script which can be used to test said Java deployment tool.

29. A computer readable medium as recited in claim 26, wherein said programming script is a tutorial script for said Java deployment tool.

30. A computer readable medium as recited in claim 26, wherein said one or more Java components are one or more WEB applications.

31. A computer readable medium as recited in claim 26, wherein said one or more Java components can be one or more of the following components: an Enterprise Java Bean, a client application, a resource adaptor, and a WEB application.

32. A computer readable medium as recited in claim 26, wherein said Java Programming Interface comprises:

a Java-based Programming Interface which can be used to simulate interactions with one or more screens which can be generated using said GUI of said Java deployment tool; and
a Java-based system layer which can operate to implement said one or more interactions.

33. A computer readable medium as recited in claim 26, wherein said at least one functionality is finding a screen or field in a screen, or entering a value in a field in a screen, said screen being capable of being generated by said GUI of said Java deployment tool.

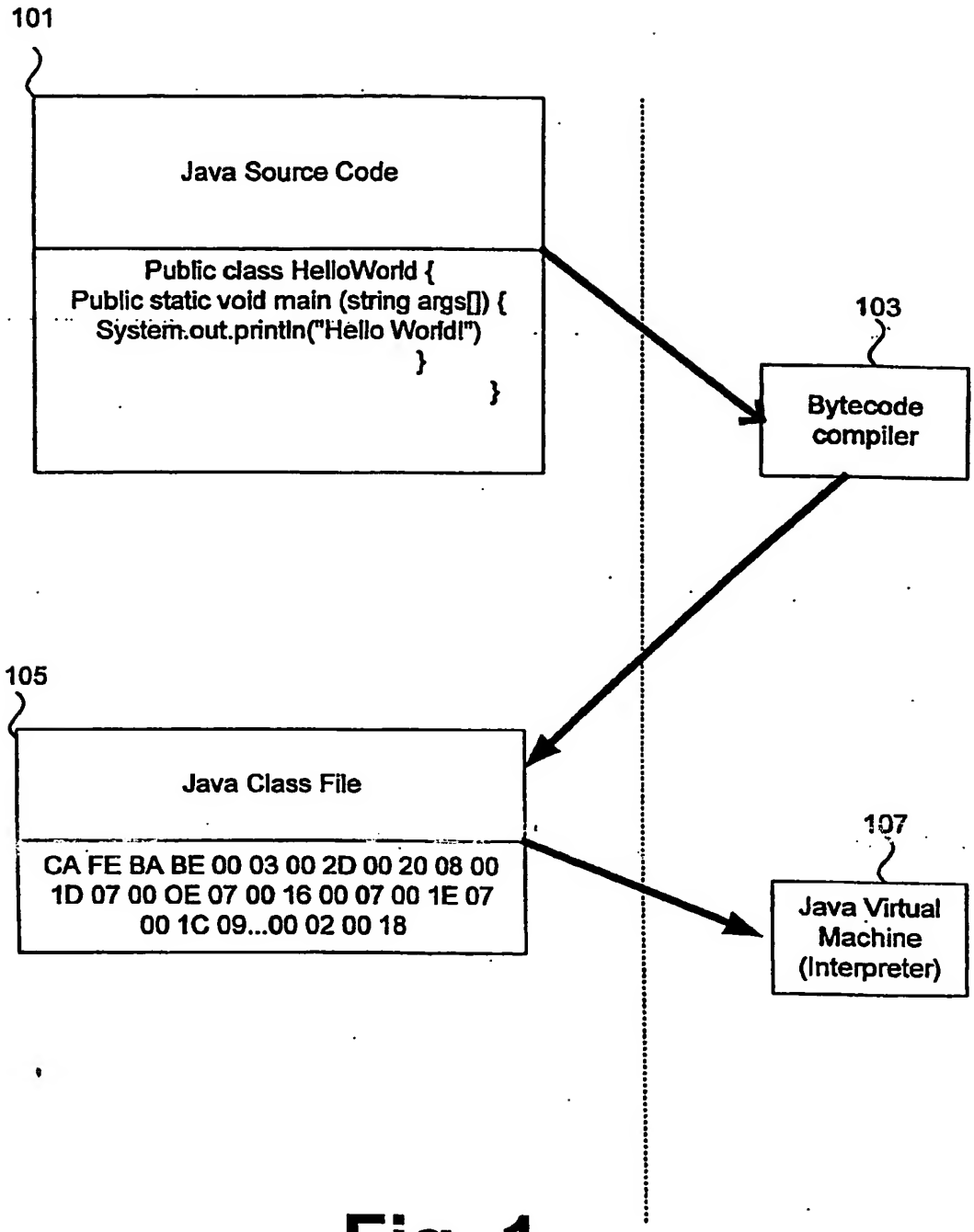
101
Java Source Code
Public class HelloWorld {
Public static void main (string args[]) {
System.out.println("Hello World!")
}
}
103
Bytecode compiler
105
Java Class File
CA FE BA BE 00 03 00 2D 00 20 08 00
1D 07 00 OE 07 00 16 00 07 00 1E 07
00 1C 09...00 02 00 18
107
Java Virtual Machine (Interpreter)

Fig. 1

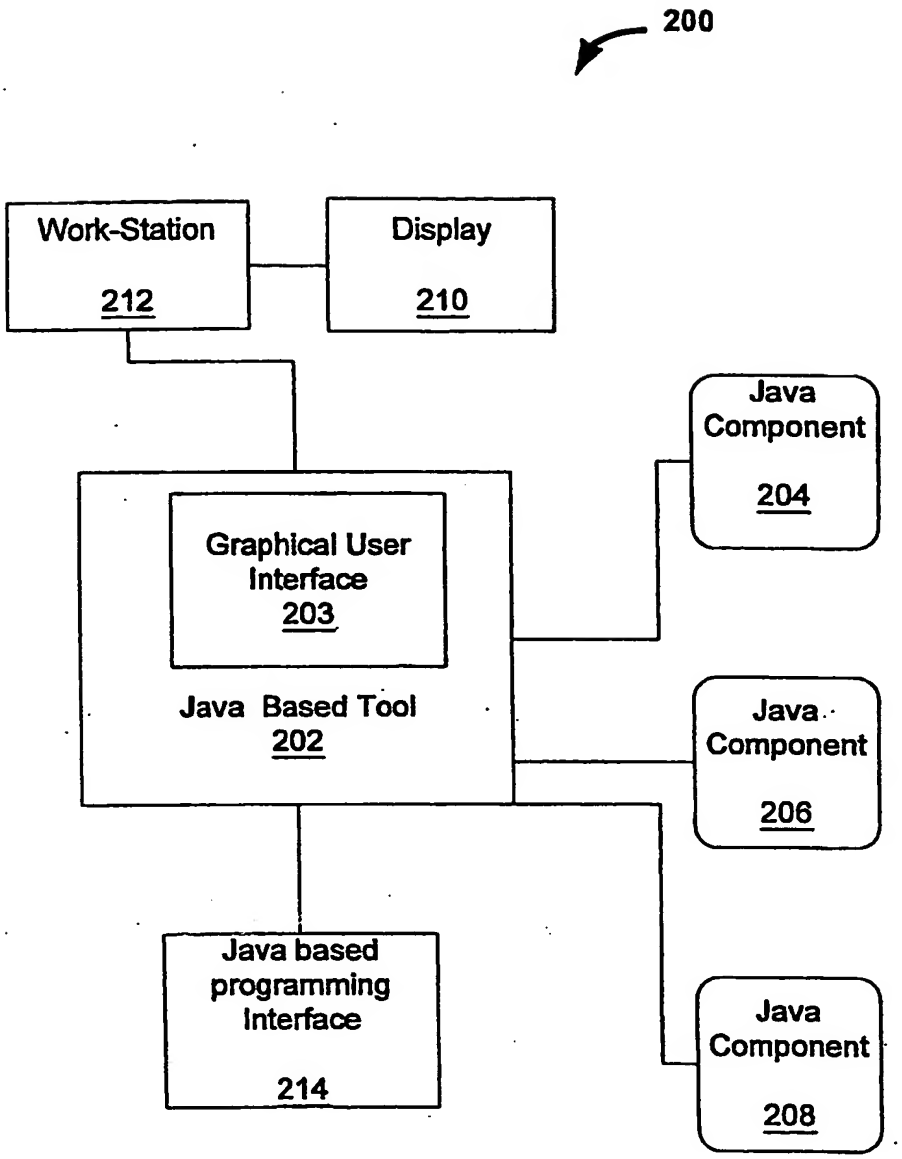
200
Work-Station
212
Display
210
Java
Component
204
Graphical User
Interface
203
Java Based Tool
202
Java
Component
206
Java based
programming
Interface
214
Java
Component
208

Fig. 2

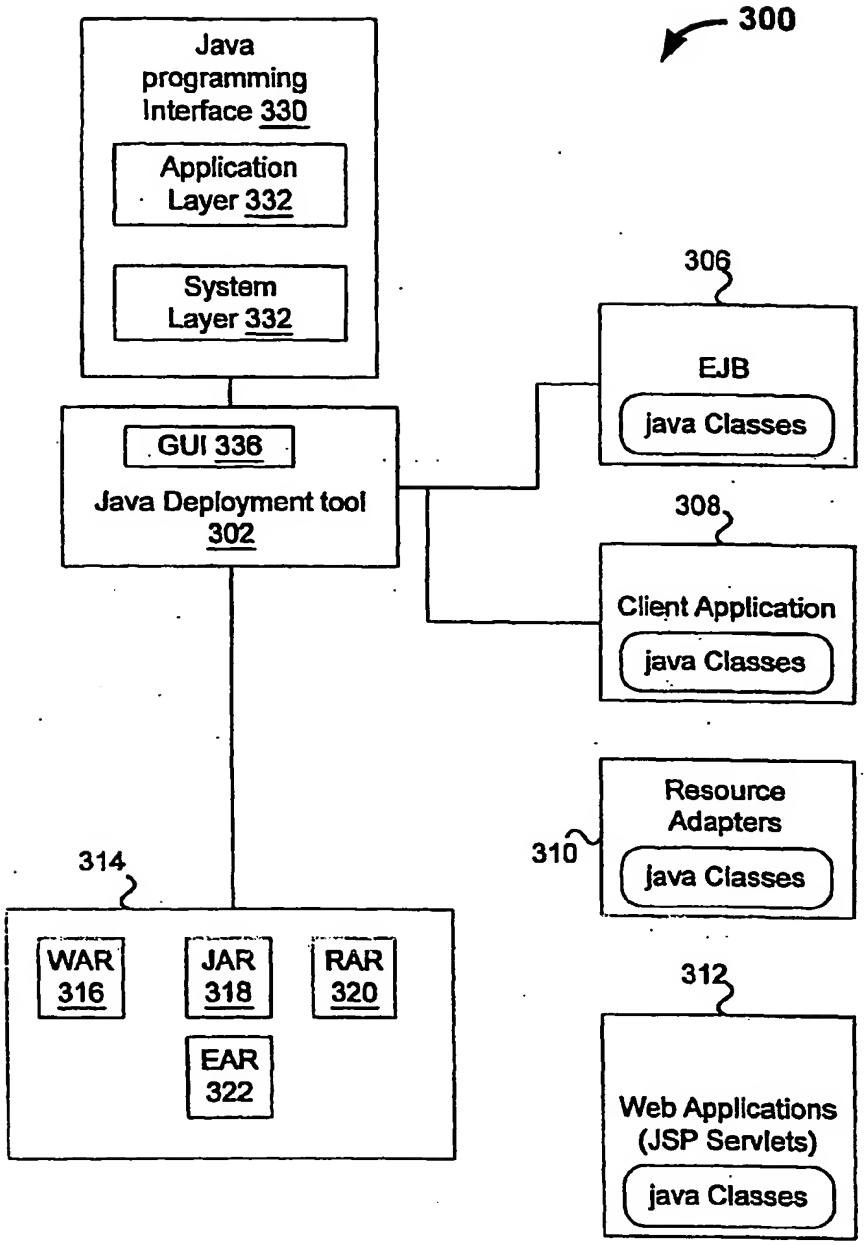
300
Java programming Interface 330
Application Layer 332
System Layer 332
GUI 336
Java Deployment tool 302
306
EJB
java Classes
308
Client Application
java Classes
Resource Adapters
310
java Classes
312
Web Applications (JSP Servlets)
java Classes
314
WAR 316
JAR 318
RAR 320
EAR 322

Fig. 3

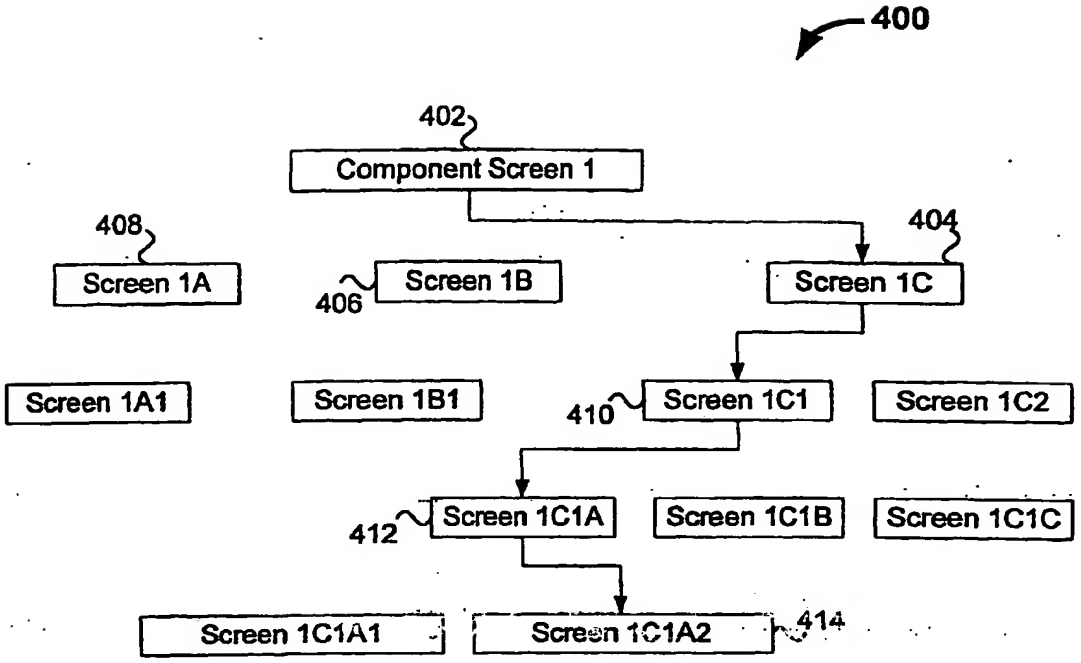
400
402
Component Screen 1
408
Screen 1A
406
Screen 1B
404
Screen 1C
Screen 1A1
Screen 1B1
410
Screen 1C1
Screen 1C2
412
Screen 1C1A
Screen 1C1B
Screen 1C1C
Screen 1C1A1
Screen 1C1A2
414

Fig. 4

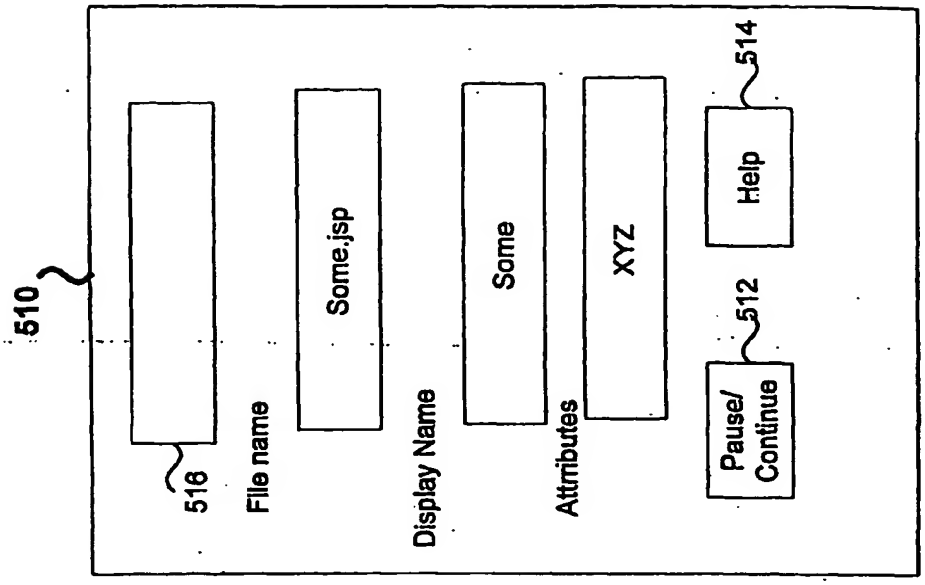
510
516
File name
Some.jsp
Display Name
Some
Attributes
XYZ
512
Pause/
Continue
514
Help

Fig. 5B

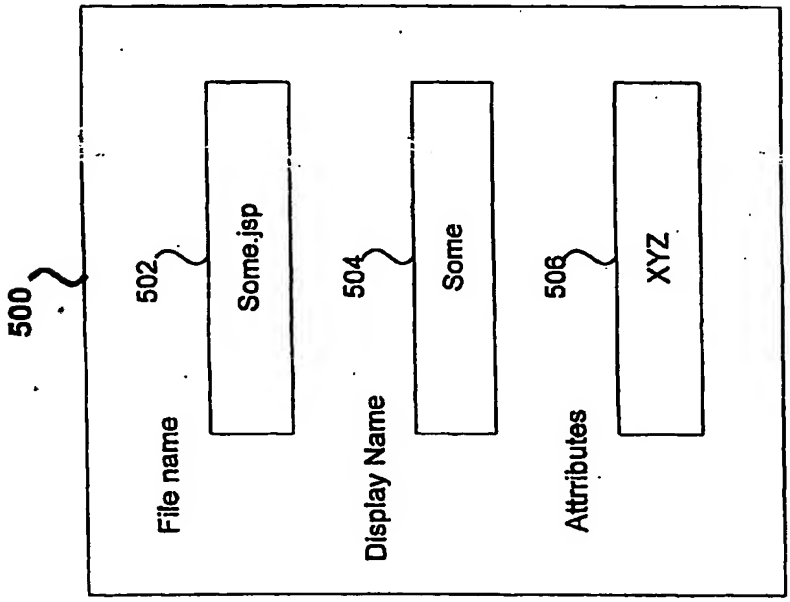
500
File name
502
Some.jsp
Display Name
504
Some
Attributes
506
XYZ

Fig. 5A

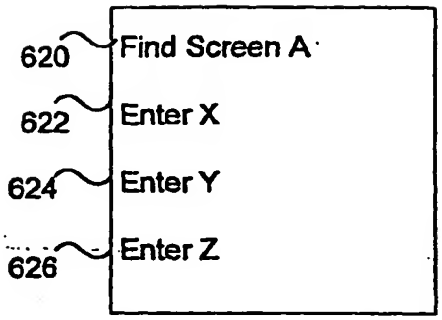
620
Find Screen A
622
Enter X
624
Enter Y
626
Enter Z

Fig. 6B

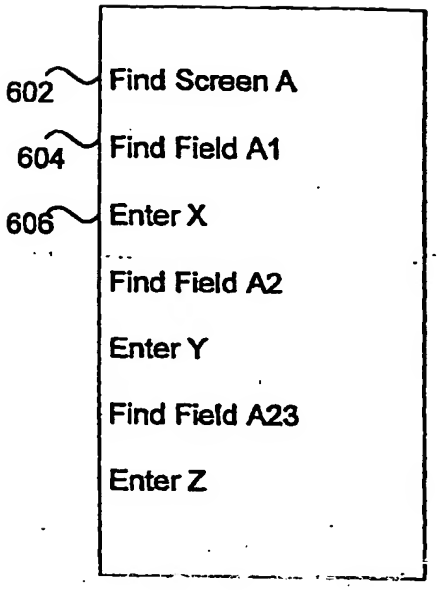
602
Find Screen A
604
Find Field A1
606
Enter X
Find Field A2
Enter Y
Find Field A23
Enter Z

Fig. 6A